COVID-19 Future Forecasting Using Supervised Machine Learning Models

Project Report

Bachelor of Technology

in

Department of Electronics and Computer Engineering

By

S. Susmitha (190050056)

V. Sowmya (190050079)

under the supervision of

Dr. A. Kiran Kumar

Assistant Professor

(DEEMED TO BE UNIVERSITY)

Koneru Lakshmaiah Education Foundation

(Deemed to be University estd., u/s 3 of UGC Act 1956)

Greenfields, Vaddeswaram, Guntur (Dist.), Andhra Pradesh - 522502

Nov, 2021

Table of Contents

Abstract:

The spread of COVID-19 in the entire world has put the humankind in danger. Machine learning (ML) based forecasting mechanisms have proved their significance to anticipate in perioperative outcomes to improve the decision making on the future course of actions. The assets of probably the biggest economies are worried because of the enormous infectivity and contagiousness of this illness.

The ability of ML models to conjecture the quantity of forthcoming patients influenced by COVID-19 which is by and by considered as a likely danger to humanity. Specifically, four standard estimating models linear regression (LR), least total shrinkage and determination administrator (LASSO) Support vector Machine (SVM) have been utilized in this examination to figure the undermining components of COVID-19.

Three types of predictions are made by each of the models, such as the number of newly infected cases, the number of deaths, and the number of recoveries in the next 10 days. The results produced by the study proves it a promising mechanism to use these methods for the current scenario of the COVID-19 pandemic. To defeat the issue, Proposed strategy utilizing the exponential smoothing (ES) anticipate the quantity of COVID-19 cases in next 30 days ahead and impact of preventive estimates like social seclusion and lockdown on the spread of COVID-19.

Introduction:

• COVID-19 was first discovered in Wuhan, China in December 2019. The World Health Organization (WHO) later declared the new emerging disease as a pandemic (Huang et al. 2020). Recent studies reported that COVID-19 is transmitted among humans by droplet infection or direct contact.

• The WHO has specified that the main human-to-human transmission mechanism varies, but still can be generalized as direct contact with an infected person through shaking hands, exposure to droplets coming out during coughing or sneezing, and by traveling to an affected area and attaining the virus in one or other way.

• The core symptoms of COVID-19 highly vary, ranging from being severely affected to being asymptomatic and the infected individuals can experience from mild to very severe respiratory illnesses. High fever, cough, sore throat and muscular pain were the primary symptoms in most of the symptomatic cases. However, severe cases suffer from pneumonia, microcoagulopathies, and septic shock. Rapid clinical deterioration of the cases can lead to death (Qiu et al. 2020; Wu et al. 2020). Mostly, old-aged people and those who have pre-existing medical conditions.

• e.g., diabetes mellitus, chronic respiratory disease, or cancer are more likely to experience manifestations and consequences of COVID-19 infection World Health Organization (WHO) (2020).

• COVID-19 is being built up and its nature and characteristics are being discovered especially, its very quick ability to change its nature evolving new variants based on its accelerated genetic mutations. Therefore, thoroughgoing observational studies are being performed to establish facts about COVID-19 to find out treatment or a vaccine that may help in ending its pandemic.

• Many research studies on COVID-19 are published and others are on the lane, and floods of huge data about it are constantly accumulating, without reaching a strong prediction about the transmission and end of the pandemic. In our current study, we deployed machine learning approaches for predicting the spread of the virus in several selected countries.

• Yet, the same approach can be applied for predicting the spread of COVID-19 infection in any other country, since the nature of the virus is nearly the same everywhere.

Requirements Elicitation:

->DATASET: Corona Virus

-> LANGUAGE: PYTHON

->ENVIRONMENT: JUPYTER NOTEBOOK(ANACONDA3)

Problem Modelling:

• Processing the dataset.

• Using Linear regression model.

• Predicting the covid cases as positive negative recovered and deaths.

• Tuning the data set.

• Using the SVM algorithm predict the number of cases for the upcoming.

• Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

• Linear Regression Model is:

• When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we process the data set.

• We have to again modify the dataset to make adjust and predict the covid cases as we above mentioned and implement the code and get the outputs.

• Here we will be using linear regression model and time series at the most we needed so dataset is corona-virus-data-set.

• Train the model by the algorithm and using with the datset acquired after all the things done above.

• Prediction the values using the SVM algorithm.

• Tune the data to get the actual summary of the covid cases of that particular day.

System Design:

Algorithm:

STEP-1: Load the dataset.

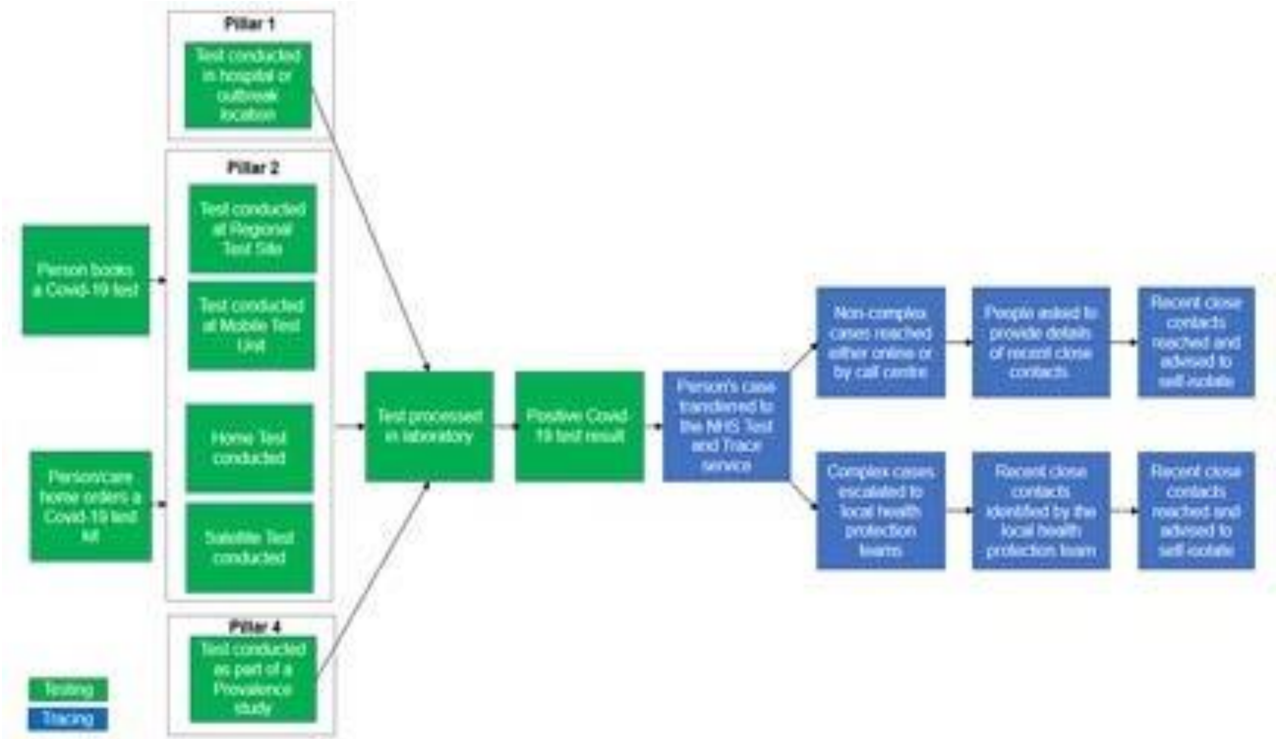
STEP-2: Import the necessary packages and Libraries.

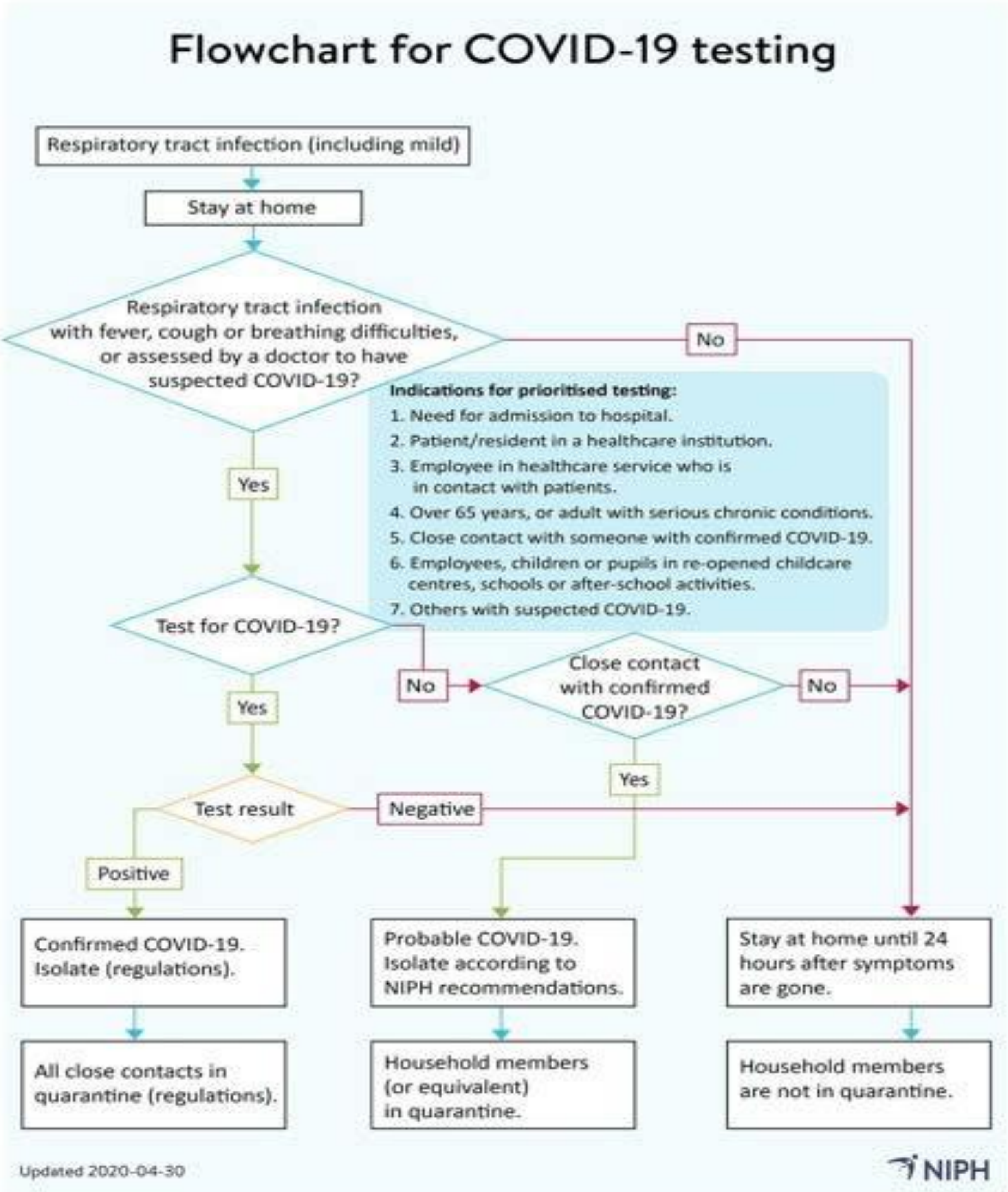
STEP-3: Do the prediction of the dataset.
{
Positive;
Negative;
Recovered;
deaths;
}

STEP-4: Stationarity check the data using techniques.

STEP-5: Predict and tune the data.

Flowchart:





Flowchart for COVID-19 testing
Respiratory tract infection (including mild)
Stay at home
Respiratory tract infection with fever, cough or breathing difficulties, or assessed by a doctor to have suspected COVID-19?
No
Yes
Indications for prioritised testing:
1. Need for admission to hospital.
2. Patient/resident in a healthcare institution.
3. Employee in healthcare service who is in contact with patients.
4. Over 65 years, or adult with serious chronic conditions.
5. Close contact with someone with confirmed COVID-19.
6. Employees, children or pupils in re-opened childcare centres, schools or after-school activities.
7. Others with suspected COVID-19.
Test for COVID-19?
No
Close contact with confirmed COVID-19?
No
Yes
Yes
Test result
Negative
Positive
Confirmed COVID-19. Isolate (regulations).
Probable COVID-19. Isolate according to NIPH recommendations.
Stay at home until 24 hours after symptoms are gone.
All close contacts in quarantine (regulations).
Household members (or equivalent) in quarantine.
Household members are not in quarantine.
Updated 2020-04-30
NIPH

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import datetime as dt
from datetime import timedelta
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from statsmodels.tsa.api import Holt
covid = pd.read_csv("covid_19_data.csv")
covid.head(10)



print("Size/Shape of the dataset",covid.shape)
print("Checking for null values",covid.isnull().sum())
print("Checking Data-type",covid.dtypes)



#Dropping the column
covid.drop(["SNo"],1,inplace=True)
covid.isnull().sum()



covid["ObservationDate"] = pd.to_datetime(covid["ObservationDate"])
#Grouping differnent types of cases as per the date
datewise =
covid.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"sum","
Deaths":"sum"})
print("Basic Information")
print("Total number of Confirmed cases around the
world",datewise["Confirmed"].iloc[-1])
print("Total number of Recovered cases around the
world",datewise["Recovered"].iloc[-1])
print("Total number of Death cases around the world",datewise["Deaths"].iloc[-1])
print("Total number of Active cases around the world",(datewise["Confirmed"].iloc[-
1]-datewise["Recovered"].iloc[-1]-datewise["Deaths"].iloc[-1]))
```

```
print("Total number of Closed cases around the
world",(datewise["Recovered"].iloc[-1]+datewise["Deaths"].iloc[-1]))



plt.figure(figsize=(15,5))
sns.barplot(x=datewise.index.date,y=datewise["Confirmed"]-datewise["Recovered"]-
datewise["Deaths"])
plt.title("Distributions plot for Active Cases")
plt.xticks(rotation=90)



plt.figure(figsize=(15,5))
sns.barplot(x=datewise.index.date,y=datewise["Recovered"]+datewise["Deaths"])
plt.title("Distribution plot for Closed Cases")
plt.xticks(rotation=90)



datewise["WeekofYear"] = datewise.index.weekofyear
week_num = []
weekwise_confirmed = []
weekwise_recovered = []
weekwise_deaths = []
w = 1
for i in list(datewise["WeekofYear"].unique()):
weekwise_confirmed.append(datewise[datewise["WeekofYear"]==i]["Confirmed"].i
loc[-1])
weekwise_recovered.append(datewise[datewise["WeekofYear"]==i]["Recovered"].il
oc[-1])
weekwise_deaths.append(datewise[datewise["WeekofYear"]==i]["Deaths"].iloc[-1])
week_num.append(w)
 w=w+1
plt.figure(figsize=(8,5))
plt.plot(week_num,weekwise_confirmed,linewidth=3)
plt.plot(week_num,weekwise_recovered,linewidth =3)
plt.plot(week_num,weekwise_deaths,linewidth = 3)
plt.xlabel("WeekNumber")
plt.ylabel("Number of cases")
plt.title("Weekly Progress of different types of cases")
```

```
fig,(ax1,ax2) = plt.subplots(1,2,figsize=(12,4))
sns.barplot(x= week_num,y=pd.Series(weekwise_confirmed).diff().fillna(0),ax=ax1)
sns.barplot(x= week_num,y=pd.Series(weekwise_deaths).diff().fillna(0),ax=ax2)
ax1.set_xlabel("Week Number")
ax2.set_xlabel("Week Number")
ax1.set_ylabel("Numberof Confirmed cases")
ax2.set_ylabel("Numberof Death cases")
ax1.set_title("Weekly increase in number of Confirmed cases")
ax2.set_title("Weekly increase in number of Death Cases")
plt.show()
```

```
print("Average increase in number of Confirmed cases
everyday:",np.round(datewise["Confirmed"].diff().fillna(0).mean()))
print("Average increase in number of Recovered cases
everyday:",np.round(datewise["Recovered"].diff().fillna(0).mean()))
print("Average increase in number of Death cases
everyday:",np.round(datewise["Deaths"].diff().fillna(0).mean()))
```

```
plt.figure(figsize=(15,6))
plt.plot(datewise["Confirmed"].diff().fillna(0),label="Daily increase in confirmed
cases",linewidth=3)
plt.plot(datewise["Recovered"].diff().fillna(0),label="Daily increase in recovered
cases",linewidth=3)
plt.plot(datewise["Deaths"].diff().fillna(0),label="Daily increase in death
cases",linewidth=3)
plt.xlabel("Timestamp")
plt.ylabel("Daily increase")
plt.title("Daily increase")
plt.legend()
plt.xticks(rotation=90)
plt.show()
```

```
#Country wise analysis
#Calculating Country wise Mortality rate
countrywise=
covid[covid["ObservationDate"]==covid["ObservationDate"].max()].groupby(["Cou
ntry/Region"]).agg({"Confirmed":"sum","Recovered":"sum","Deaths":"sum"}).sort_
```

```
values(["Confirmed"],ascending=False)
countrywise["Mortality"]=(countrywise["Deaths"]/countrywise["Recovered"])*100
countrywise["Recovered"]=(countrywise["Recovered"]/countrywise["Confirmed"])*
100



fig,(ax1,ax2)=plt.subplots(1,2,figsize=(25,10))
top_15confirmed =
countrywise.sort_values(["Confirmed"],ascending=False).head(15)
top_15deaths = countrywise.sort_values(["Deaths"],ascending=False).head(15)
sns.barplot(x=top_15confirmed["Confirmed"],y=top_15confirmed.index,ax=ax1)
ax1.set_title("Top 15 countries as per number of confirmed cases")
sns.barplot(x=top_15deaths["Deaths"],y=top_15deaths.index,ax=ax2)
ax1.set_title("Top 15 countries as per number of death cases")



#Data Anlaysis for India
india_data = covid[covid["Country/Region"]=="India"]
datewise_india =
india_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"su
m","Deaths":"sum"})
print(datewise_india.iloc[-1])
print("Total Active Cases",datewise_india["Confirmed"].iloc[-1]-
datewise_india["Recovered"].iloc[-1]-datewise_india["Deaths"].iloc[-1])
print("Total Closed Cases",datewise_india["Recovered"].iloc[-
1]+datewise_india["Deaths"].iloc[-1])



#Data Anlaysis for US
us_data = covid[covid["Country/Region"]=="US"]
datewise_us =
us_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"sum"
,"Deaths":"sum"})
print(datewise_us.iloc[-1])
print("Total Active Cases",datewise_us["Confirmed"].iloc[-1]-
datewise_us["Recovered"].iloc[-1]-datewise_us["Deaths"].iloc[-1])
print("Total Closed Cases",datewise_us["Recovered"].iloc[-
1]+datewise_us["Deaths"].iloc[-1])
```

```
datewise_india["WeekofYear"] = datewise_india.index.weekofyear
week_num_india = []
india_weekwise_confirmed = []
india_weekwise_recovered = []
india_weekwise_deaths = []
w = 1
for i in list(datewise_india["WeekofYear"].unique()):
india_weekwise_confirmed.append(datewise_india[datewise_india["WeekofYear"]=
=i]["Confirmed"].iloc[-1])
india_weekwise_recovered.append(datewise_india[datewise_india["WeekofYear"]=
=i]["Recovered"].iloc[-1])
india_weekwise_deaths.append(datewise_india[datewise_india["WeekofYear"]==i][
"Deaths"].iloc[-1])
 week_num_india.append(w)
 w=w+1
plt.figure(figsize=(8,5))
plt.plot(week_num_india,india_weekwise_confirmed,linewidth=3)
plt.plot(week_num_india,india_weekwise_recovered,linewidth =3)
plt.plot(week_num_india,india_weekwise_deaths,linewidth = 3)
plt.xlabel("WeekNumber")
plt.ylabel("Number of cases")
plt.title("Weekly Progress of different types of cases")
```

```
max_ind = datewise_india["Confirmed"].max()
china_data = covid[covid["Country/Region"]=="Mainland China"]
Italy_data = covid[covid["Country/Region"]=="Italy"]
US_data = covid[covid["Country/Region"]=="US"]
spain_data = covid[covid["Country/Region"]=="Spain"]
datewise_china =
china_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"su
m","Deaths":"sum"})
datewise_Italy =
Italy_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"su
m","Deaths":"sum"})
datewise_US=US_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Re
covered":"sum","Deaths":"sum"})
datewise_Spain=spain_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum"
,"Recovered":"sum","Deaths":"sum"})
print("It took",datewise_india[datewise_india["Confirmed"]>0].shape[0],"days in
India to reach",max_ind,"Confirmed Cases")
print("It
```

```
took",datewise_Italy[(datewise_Italy["Confirmed"]>0)&(datewise_Italy["Confirmed
"]<=max_ind)].shape[0],"days in Italy to reach number of Confirmed Cases")
print("It
took",datewise_US[(datewise_US["Confirmed"]>0)&(datewise_US["Confirmed"]<=
max_ind)].shape[0],"days in US to reach number of Confirmed Cases")
print("It
took",datewise_Spain[(datewise_Spain["Confirmed"]>0)&(datewise_Spain["Confir
med"]<=max_ind)].shape[0],"days in Spain to reach number of Confirmed Cases")
print("It
took",datewise_china[(datewise_china["Confirmed"]>0)&(datewise_china["Confirm
ed"]<=max_ind)].shape[0],"days in China to reach number of Confirmed Cases")
```

```
datewise["Days Since"]=datewise.index-datewise.index[0]
datewise["Days Since"] = datewise["Days Since"].dt.days
train_ml = datewise.iloc[:int(datewise.shape[0]*0.95)]
valid_ml = datewise.iloc[:int(datewise.shape[0]*0.95):]
model_scores=[]
lin_reg = LinearRegression(normalize=True)
svm = SVR(C=1,degree=5,kernel='poly',epsilon=0.001)
lin_reg.fit(np.array(train_ml["Days Since"]).reshape(-
1,1),np.array(train_ml["Confirmed"]).reshape(-1,1))
svm.fit(np.array(train_ml["Days Since"]).reshape(
1,1),np.array(train_ml["Confirmed"]).reshape(-1,1))
```

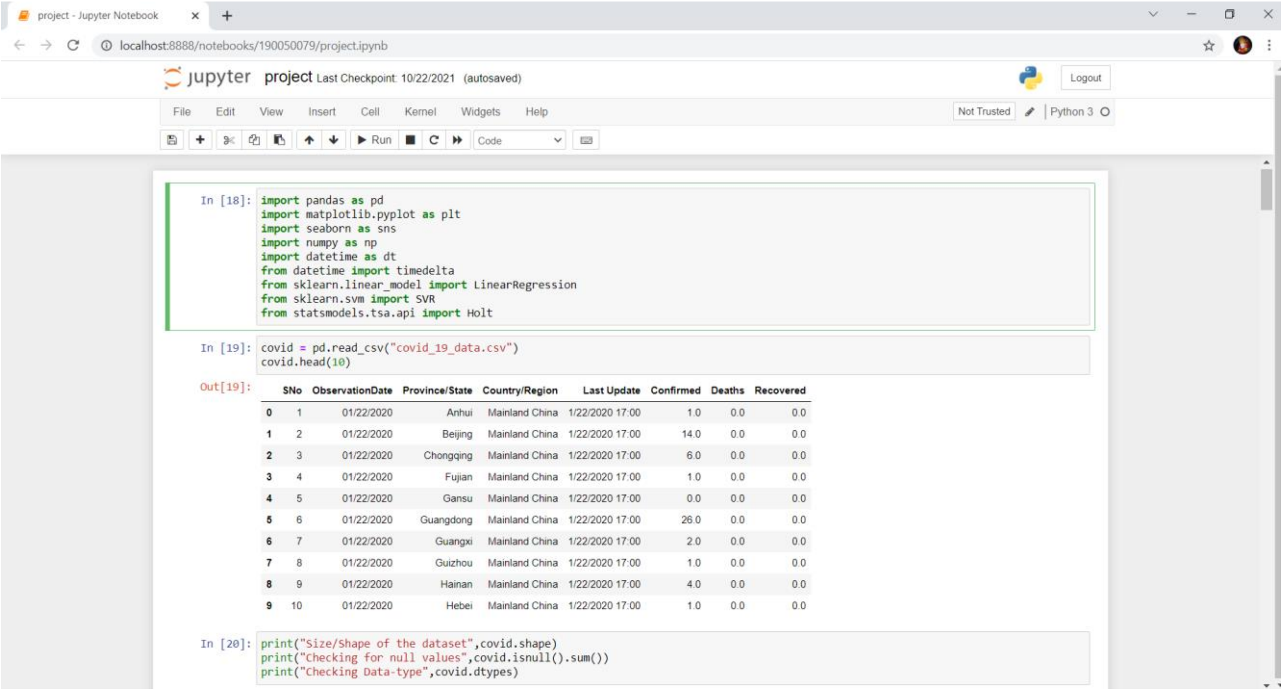
```
prediction_valid_lin_reg = lin_reg.predict(np.array(valid_ml["Days
Since"]).reshape(-1,1))
prediction_valid_svm = svm.predict(np.array(valid_ml["Days Since"]).reshape(-1,1))
new_date = []
new_prediction_lr=[]
new_prediction_svm=[]
for i in range(1,18):
 new_date.append(datewise.index[-1]+timedelta(days=i))
new_prediction_lr.append(lin_reg.predict(np.array(datewise["Days
Since"].max()+i).reshape(-1,1))[0][0])
new_prediction_svm.append(svm.predict(np.array(datewise["Days
Since"].max()+i).reshape(-1,1))[0])
pd.set_option("display.float_format",lambda x: '%.f' % x)
model_predictions=pd.DataFrame(zip(new_date,new_prediction_lr,new_prediction_
svm),columns = ["Dates","LR","SVR"])
```

```
model_predictions.head(5)



model_train=datewise.iloc[:int(datewise.shape[0]*0.85)]
valid=datewise.iloc[int(datewise.shape[0]*0.85):]
holt=Holt(np.asarray(model_train["Confirmed"])).fit(smoothing_level=1.4,smoothing_slope=0.2)
y_pred = valid.copy()
y_pred["Holt"]=holt.forecast(len(valid))
holt_new_date=[]
holt_new_prediction=[]
for i in range(1,18):
holt_new_date.append(datewise.index[-1]+timedelta(days=i))
holt_new_prediction.append(holt.forecast((len(valid)+i))[-1])
model_predictions["Holts Linear Model Prediction"]=holt_new_prediction
model_predictions.head()
```

Implementation:



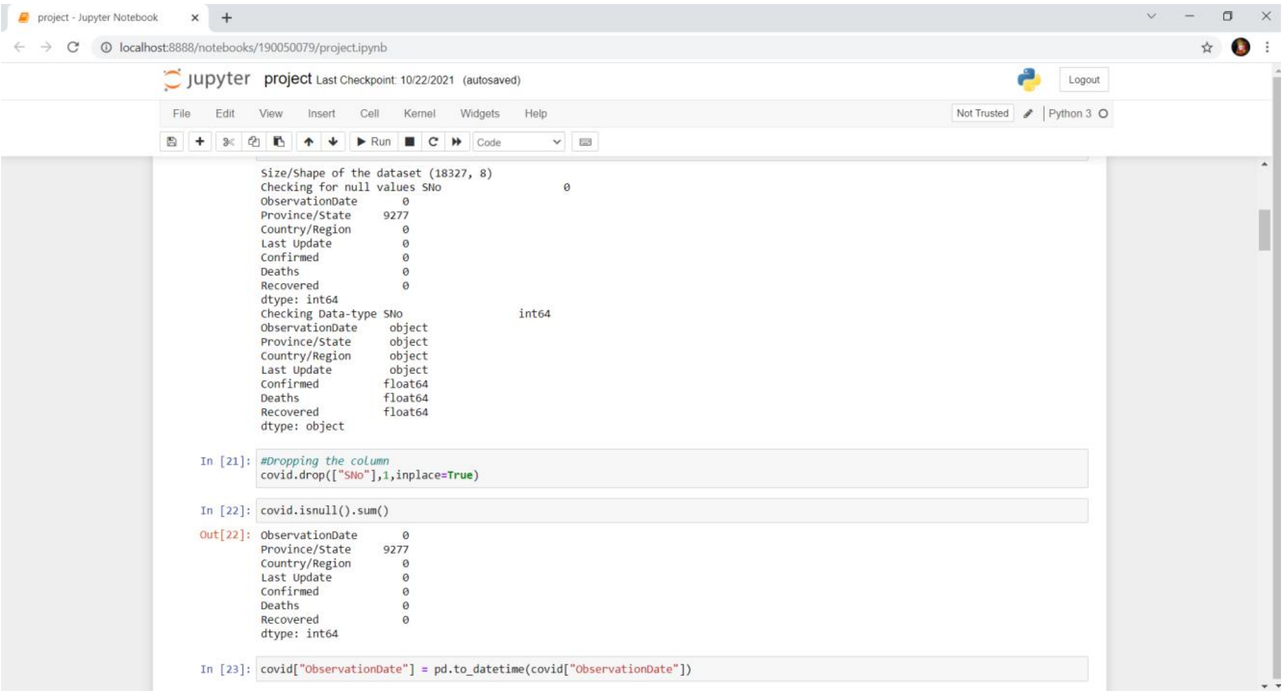

```
In [18]: import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
         import numpy as np
         import datetime as dt
         from datetime import timedelta
         from sklearn.linear_model import LinearRegression
         from sklearn.svm import SVR
         from statsmodels.tsa.api import Holt

In [19]: covid = pd.read_csv("covid_19_data.csv")
         covid.head(10)
```

Out[19]:

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
0	1	01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
1	2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0
2	3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0
3	4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
4	5	01/22/2020	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0
5	6	01/22/2020	Guangdong	Mainland China	1/22/2020 17:00	26.0	0.0	0.0
6	7	01/22/2020	Guangxi	Mainland China	1/22/2020 17:00	2.0	0.0	0.0
7	8	01/22/2020	Guizhou	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
8	9	01/22/2020	Hainan	Mainland China	1/22/2020 17:00	4.0	0.0	0.0
9	10	01/22/2020	Hebei	Mainland China	1/22/2020 17:00	1.0	0.0	0.0

```
In [20]: print("Size/Shape of the dataset",covid.shape)
         print("Checking for null values",covid.isnull().sum())
         print("Checking Data-type",covid.dtypes)
```




```
Size/Shape of the dataset (18327, 8)
Checking for null values SNo                0
ObservationDate       0
Province/State     9277
Country/Region        0
Last Update           0
Confirmed             0
Deaths                0
Recovered             0
dtype: int64
Checking Data-type SNo                 int64
ObservationDate     object
Province/State      object
Country/Region      object
Last Update         object
Confirmed          float64
Deaths             float64
Recovered          float64
dtype: object

In [21]: #Dropping the column
         covid.drop(["SNo"],1,inplace=True)

In [22]: covid.isnull().sum()

Out[22]: ObservationDate       0
         Province/State     9277
         Country/Region        0
         Last Update           0
         Confirmed             0
         Deaths                0
         Recovered             0
         dtype: int64

In [23]: covid["ObservationDate"] = pd.to_datetime(covid["ObservationDate"])
```

```
In [24]: #Grouping differnent types of cases as per the date
datewise = covid.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"sum","Deaths":"sum"})
```

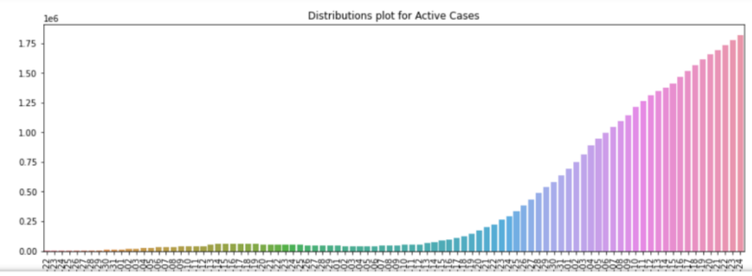
```
In [25]: print("Basic Information")
print("Total number of Confirmed cases around the world",datewise["Confirmed"].iloc[-1])
print("Total number of Recovered cases around the world",datewise["Recovered"].iloc[-1])
print("Total number of Death cases around the world",datewise["Deaths"].iloc[-1])
print("Total number of Active cases around the world",(datewise["Confirmed"].iloc[-1]-datewise["Recovered"].iloc[-1]-datewise["De
print("Total number of Closed cases around the world",(datewise["Recovered"].iloc[-1]+datewise["Deaths"].iloc[-1]))
```

```
Basic Information
Total number of Confirmed cases around the world 2811193.0
Total number of Recovered cases around the world 793601.0
Total number of Death cases around the world 197159.0
Total number of Active cases around the world 1820433.0
Total number of Closed cases around the world 990760.0
```

```
In [26]: plt.figure(figsize=(15,5))
sns.barplot(x=datewise.index.date,y=datewise["Confirmed"]-datewise["Recovered"]-datewise["Deaths"])
plt.title("Distributions plot for Active Cases")
plt.xticks(rotation=90)
```

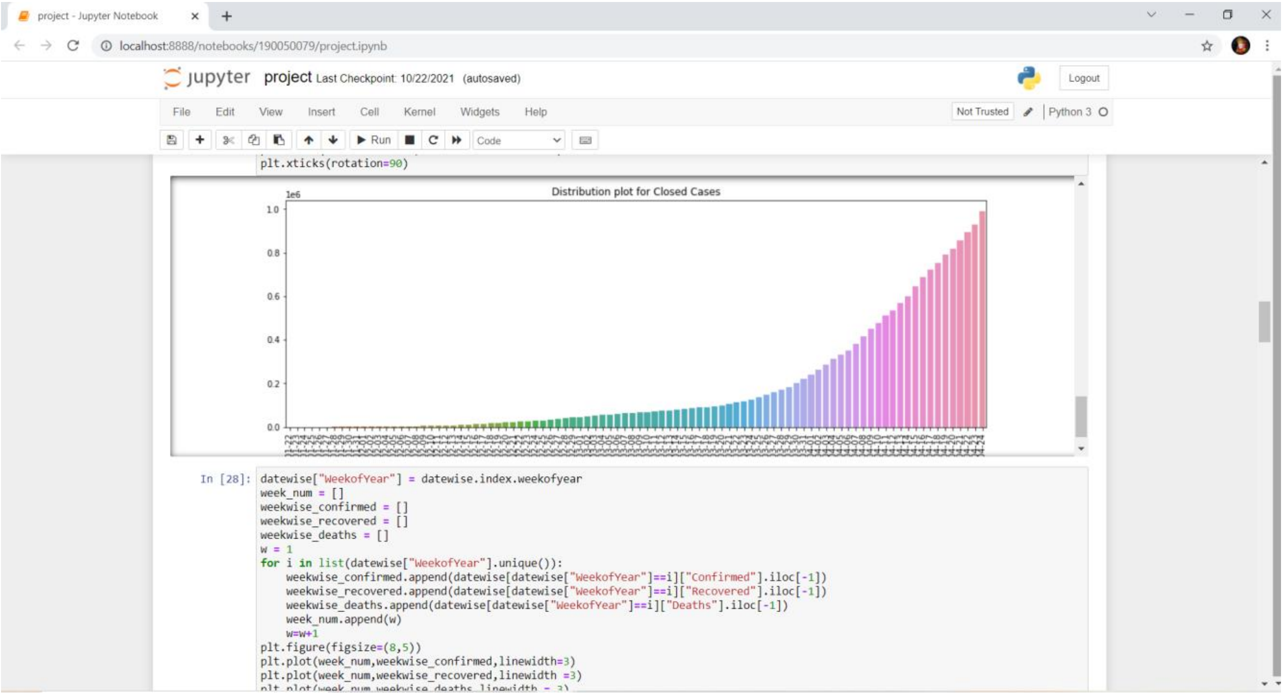
```
Out[26]: (array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
        51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
        68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
        85, 86, 87, 88, 89, 90, 91, 92, 93]),
 [Text(0, 0, '2020-01-22'),
  Text(1, 0, '2020-01-23'),
  Text(2, 0, '2020-01-24'),
  Text(3, 0, '2020-01-25'),
  Text(4, 0, '2020-01-26'),
  Text(5, 0, '2020-01-27'),
```

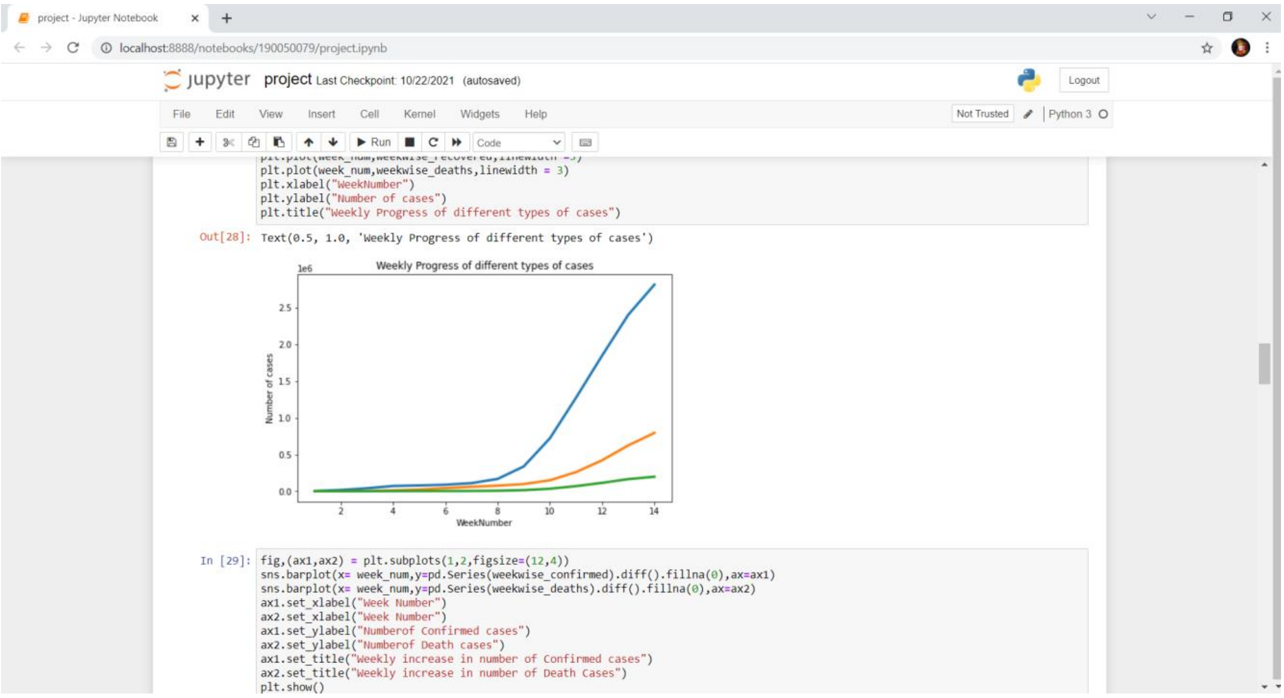
```
plt.title("Distributions plot for Active Cases")
plt.xticks(rotation=90)
```




```
In [27]: plt.figure(figsize=(15,5))
sns.barplot(x=datewise.index.date,y=datewise["Recovered"]+datewise["Deaths"])
plt.title("Distribution plot for Closed Cases")
plt.xticks(rotation=90)
```

```
Out[27]: (array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
        51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
        68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
        85, 86, 87, 88, 89, 90, 91, 92, 93]),
 [Text(0, 0, '2020-01-22'),
  Text(1, 0, '2020-01-23'),
  Text(2, 0, '2020-01-24'),
  Text(3, 0, '2020-01-25'),
```


project - Jupyter Notebook
localhost:8888/notebooks/190050079/project.ipynb
jupyter project Last Checkpoint: 10/22/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
plt.xticks(rotation=90)
Distribution plot for Closed Cases
1e6
In [28]:
datewise["WeekofYear"] = datewise.index.weekofyear
week_num = []
weekwise_confirmed = []
weekwise_recovered = []
weekwise_deaths = []
w = 1
for i in list(datewise["WeekofYear"].unique()):
weekwise_confirmed.append(datewise[datewise["WeekofYear"]==i]["Confirmed"].iloc[-1])
weekwise_recovered.append(datewise[datewise["WeekofYear"]==i]["Recovered"].iloc[-1])
weekwise_deaths.append(datewise[datewise["WeekofYear"]==i]["Deaths"].iloc[-1])
week_num.append(w)
w=w+1
plt.figure(figsize=(8,5))
plt.plot(week_num,weekwise_confirmed,linewidth=3)
plt.plot(week_num,weekwise_recovered,linewidth =3)


project - Jupyter Notebook
localhost:8888/notebooks/190050079/project.ipynb
jupyter project Last Checkpoint: 10/22/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
plt.plot(week_num,weekwise_deaths,linewidth = 3)
plt.xlabel("WeekNumber")
plt.ylabel("Number of cases")
plt.title("Weekly Progress of different types of cases")
Out[28]: Text(0.5, 1.0, 'Weekly Progress of different types of cases')
Weekly Progress of different types of cases
1e6
Number of cases
WeekNumber
In [29]:
fig,(ax1,ax2) = plt.subplots(1,2,figsize=(12,4))
sns.barplot(x= week_num,y=pd.Series(weekwise_confirmed).diff().fillna(0),ax=ax1)
sns.barplot(x= week_num,y=pd.Series(weekwise_deaths).diff().fillna(0),ax=ax2)
ax1.set_xlabel("Week Number")
ax2.set_xlabel("Week Number")
ax1.set_ylabel("Numberof Confirmed cases")
ax2.set_ylabel("Numberof Death cases")
ax1.set_title("Weekly increase in number of Confirmed cases")
ax2.set_title("Weekly increase in number of Death Cases")
plt.show()

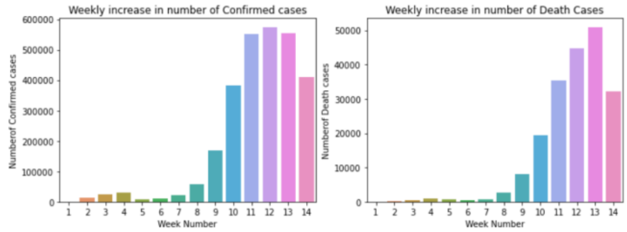
project - Jupyter Notebook

localhost:8888/notebooks/190050079/project.ipynb

jupyter project Last Checkpoint: 10/22/2021 (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

```
ax2.set_title("Weekly increase in number of Death Cases")
plt.show()
```




```
In [30]: print("Average increase in number of Confirmed cases everyday:",np.round(datewise["Confirmed"].diff().fillna(0).mean()))
print("Average increase in number of Recovered cases everyday:",np.round(datewise["Recovered"].diff().fillna(0).mean()))
print("Average increase in number of Death cases everyday:",np.round(datewise["Deaths"].diff().fillna(0).mean()))

plt.figure(figsize=(15,6))
plt.plot(datewise["Confirmed"].diff().fillna(0),label="Daily increase in confirmed cases",linewidth=3)
plt.plot(datewise["Recovered"].diff().fillna(0),label="Daily increase in recovered cases",linewidth=3)
plt.plot(datewise["Deaths"].diff().fillna(0),label="Daily increase in death cases",linewidth=3)
plt.xlabel("Timestamp")
plt.ylabel("Daily increase")
plt.title("Daily increase")
plt.legend()
plt.xticks(rotation=90)
plt.show()
```

```
Average increase in number of Confirmed cases everyday: 29900.0
Average increase in number of Recovered cases everyday: 8442.0
```

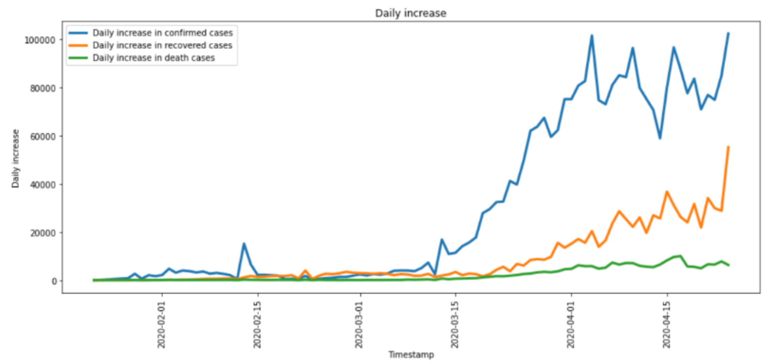
project - Jupyter Notebook

localhost:8888/notebooks/190050079/project.ipynb

jupyter project Last Checkpoint: 10/22/2021 (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

```
Average increase in number of Recovered cases everyday: 8442.0
Average increase in number of Death cases everyday: 2097.0
```




```
In [31]: #Country wise analysis
#Calculating Country wise Mortality rate
countrywise= covid[covid["ObservationDate"]==covid["ObservationDate"].max()].groupby(["Country/Region"]).agg({"Confirmed":"sum",'
countrywise["Mortality"]=(countrywise["Deaths"]/countrywise["Recovered"])*100
countrywise["Recovered"]=(countrywise["Recovered"]/countrywise["Confirmed"])*100
```

```
In [32]: fig,(ax1,ax2)=plt.subplots(1,2,figsize=(25,10))
```

project - Jupyter Notebook

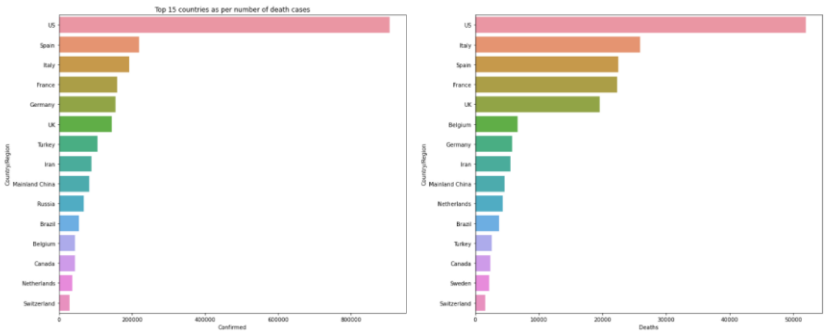
localhost:8888/notebooks/190050079/project.ipynb

jupyter project Last Checkpoint: 10/22/2021 (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

In [32]:
```
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(25,10))
top_15confirmed = countrywise.sort_values(["Confirmed"],ascending=False).head(15)
top_15deaths = countrywise.sort_values(["Deaths"],ascending=False).head(15)
sns.barplot(x=top_15confirmed["Confirmed"],y=top_15confirmed.index,ax=ax1)
ax1.set_title("Top 15 countries as per number of confirmed cases")
sns.barplot(x=top_15deaths["Deaths"],y=top_15deaths.index,ax=ax2)
ax1.set_title("Top 15 countries as per number of death cases")
```

Out[32]:
```
Text(0.5, 1.0, 'Top 15 countries as per number of death cases')
```



In [33]:
```
Anlaysis for India
data = covid[covid["Country/Region"]=="India"]
se_india = india_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"sum","Deaths":"sum"})
```


project - Jupyter Notebook

localhost:8888/notebooks/190050079/project.ipynb

jupyter project Last Checkpoint: 10/22/2021 (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

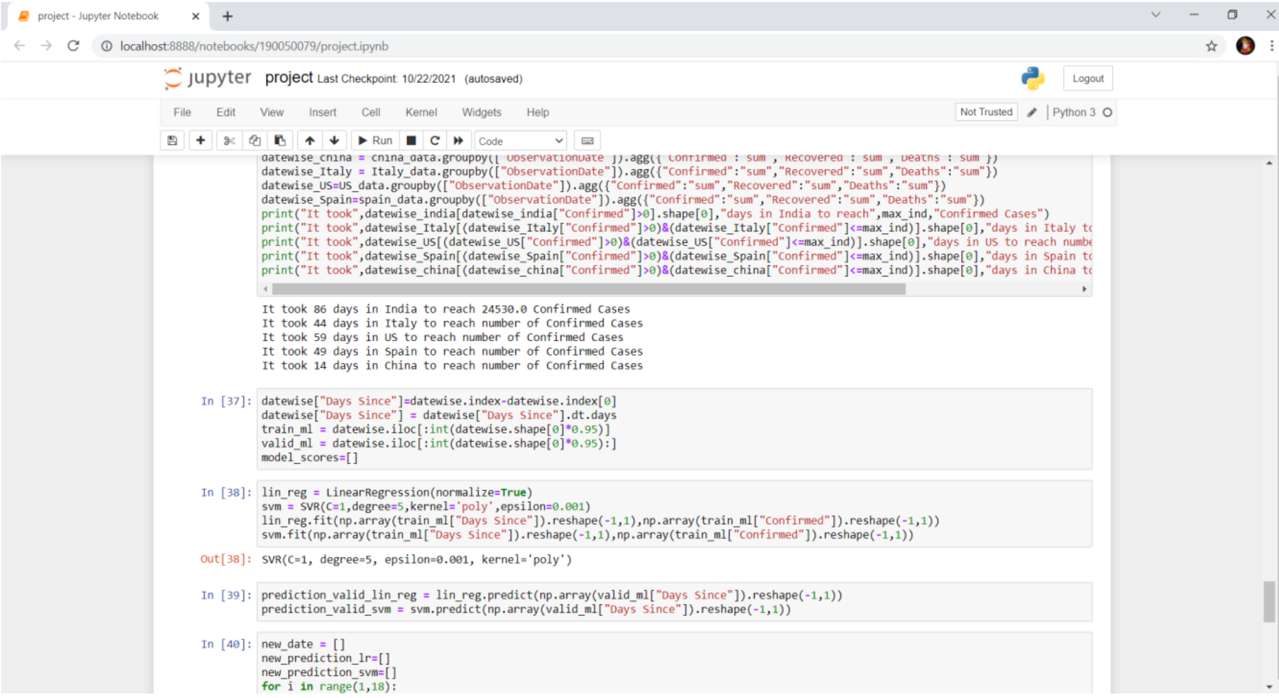

```
datewise_india.iloc[-1])
'Total Active Cases",datewise_india["Confirmed"].iloc[-1]-datewise_india["Recovered"].iloc[-1]-datewise_india["Deaths"].iloc[-1])
'Total Closed Cases",datewise_india["Recovered"].iloc[-1]+datewise_india["Deaths"].iloc[-1])
```

```
Confirmed     24530.0
Recovered      5498.0
Deaths          780.0
Name: 2020-04-24 00:00:00, dtype: float64
Total Active Cases 18252.0
Total Closed Cases 6278.0
```

In [34]:
```
#Data Anlaysis for US
us_data = covid[covid["Country/Region"]=="US"]
datewise_us = us_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"sum","Deaths":"sum"})
print(datewise_us.iloc[-1])
print("Total Active Cases",datewise_us["Confirmed"].iloc[-1]-datewise_us["Recovered"].iloc[-1]-datewise_us["Deaths"].iloc[-1])
print("Total Closed Cases",datewise_us["Recovered"].iloc[-1]+datewise_us["Deaths"].iloc[-1])
```

```
Confirmed    905333.0
Recovered     99079.0
Deaths        51949.0
Name: 2020-04-24 00:00:00, dtype: float64
Total Active Cases 754305.0
Total Closed Cases 151028.0
```

In [35]:
```
datewise_india["WeekofYear"] = datewise_india.index.weekofyear
week_num_india = []
india_weekwise_confirmed = []
india_weekwise_recovered = []
india_weekwise_deaths = []
w = 1
for i in list(datewise_india["WeekofYear"].unique()):
    india_weekwise_confirmed.append(datewise_india[datewise_india["WeekofYear"]==i]["Confirmed"].iloc[-1])
    india_weekwise_recovered.append(datewise_india[datewise_india["WeekofYear"]==i]["Recovered"].iloc[-1])
    india_weekwise_deaths.append(datewise_india[datewise_india["WeekofYear"]==i]["Deaths"].iloc[-1])
    week_num_india.append(w)
```



```
datewise_china = china_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"sum","Deaths":"sum"})
datewise_Italy = Italy_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"sum","Deaths":"sum"})
datewise_US=US_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"sum","Deaths":"sum"})
datewise_Spain=spain_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"sum","Deaths":"sum"})
print("It took",datewise_india[datewise_india["Confirmed"]>0].shape[0],"days in India to reach",max_ind,"Confirmed Cases")
print("It took",datewise_Italy[(datewise_Italy["Confirmed"]>0)&(datewise_Italy["Confirmed"]<=max_ind)].shape[0],"days in Italy to
print("It took",datewise_US[(datewise_US["Confirmed"]>0)&(datewise_US["Confirmed"]<=max_ind)].shape[0],"days in US to reach numbe
print("It took",datewise_Spain[(datewise_Spain["Confirmed"]>0)&(datewise_Spain["Confirmed"]<=max_ind)].shape[0],"days in Spain to
print("It took",datewise_china[(datewise_china["Confirmed"]>0)&(datewise_china["Confirmed"]<=max_ind)].shape[0],"days in China to

It took 86 days in India to reach 24530.0 Confirmed Cases
It took 44 days in Italy to reach number of Confirmed Cases
It took 59 days in US to reach number of Confirmed Cases
It took 49 days in Spain to reach number of Confirmed Cases
It took 14 days in China to reach number of Confirmed Cases
```

In [37]:

```
datewise["Days Since"]=datewise.index-datewise.index[0]
datewise["Days Since"] = datewise["Days Since"].dt.days
train_ml = datewise.iloc[:int(datewise.shape[0]*0.95)]
valid_ml = datewise.iloc[:int(datewise.shape[0]*0.95):]
model_scores=[]
```

In [38]:

```
lin_reg = LinearRegression(normalize=True)
svm = SVR(C=1,degree=5,kernel='poly',epsilon=0.001)
lin_reg.fit(np.array(train_ml["Days Since"]).reshape(-1,1),np.array(train_ml["Confirmed"]).reshape(-1,1))
svm.fit(np.array(train_ml["Days Since"]).reshape(-1,1),np.array(train_ml["Confirmed"]).reshape(-1,1))
```

Out[38]:

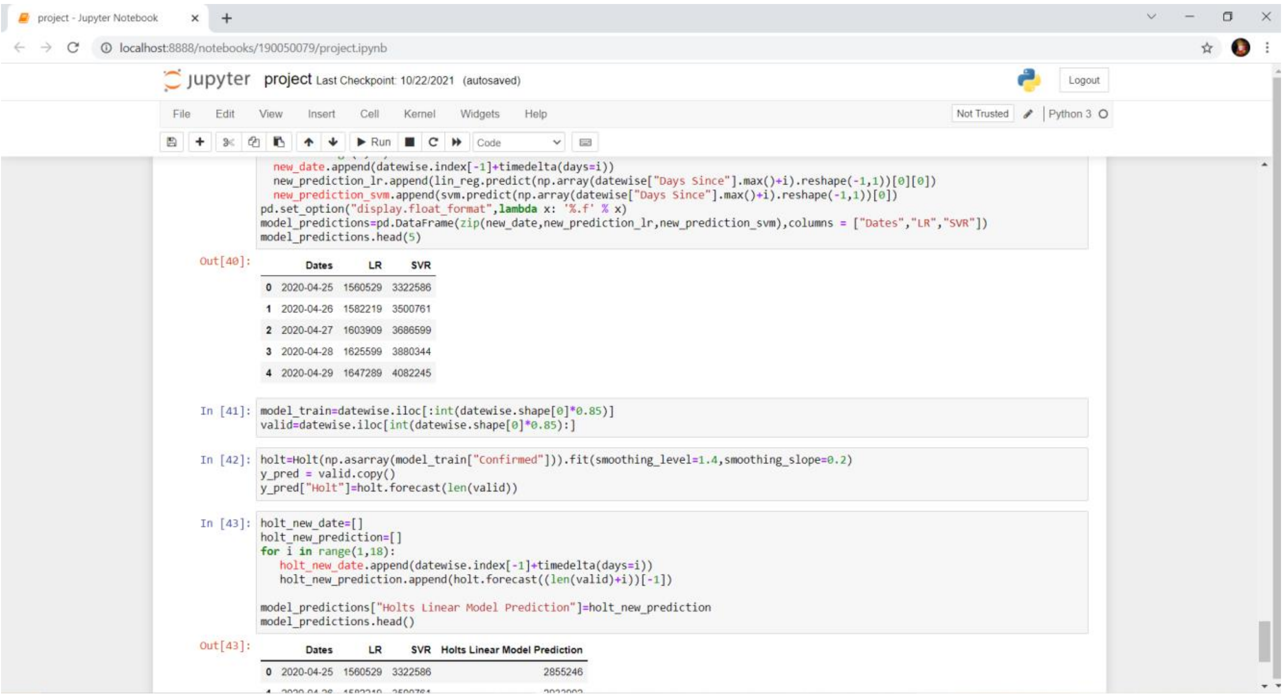
```
SVR(C=1, degree=5, epsilon=0.001, kernel='poly')
```

In [39]:

```
prediction_valid_lin_reg = lin_reg.predict(np.array(valid_ml["Days Since"]).reshape(-1,1))
prediction_valid_svm = svm.predict(np.array(valid_ml["Days Since"]).reshape(-1,1))
```

In [40]:

```
new_date = []
new_prediction_lr=[]
new_prediction_svm=[]
for i in range(1,18):
```



```
  new_date.append(datewise.index[-1]+timedelta(days=i))
  new_prediction_lr.append(lin_reg.predict(np.array(datewise["Days Since"].max()+i).reshape(-1,1))[0][0])
  new_prediction_svm.append(svm.predict(np.array(datewise["Days Since"].max()+i).reshape(-1,1))[0])
pd.set_option("display.float_format",lambda x: '%.f' % x)
model_predictions=pd.DataFrame(zip(new_date,new_prediction_lr,new_prediction_svm),columns = ["Dates","LR","SVR"])
model_predictions.head(5)
```

Out[40]:

	Dates	LR	SVR
0	2020-04-25	1560529	3322586
1	2020-04-26	1582219	3500761
2	2020-04-27	1603909	3686599
3	2020-04-28	1625599	3880344
4	2020-04-29	1647289	4082245

In [41]:

```
model_train=datewise.iloc[:int(datewise.shape[0]*0.85)]
valid=datewise.iloc[int(datewise.shape[0]*0.85):]
```

In [42]:

```
holt=Holt(np.asarray(model_train["Confirmed"])).fit(smoothing_level=1.4,smoothing_slope=0.2)
y_pred = valid.copy()
y_pred["Holt"]=holt.forecast(len(valid))
```

In [43]:

```
holt_new_date=[]
holt_new_prediction=[]
for i in range(1,18):
   holt_new_date.append(datewise.index[-1]+timedelta(days=i))
   holt_new_prediction.append(holt.forecast((len(valid)+i))[-1])

model_predictions["Holts Linear Model Prediction"]=holt_new_prediction
model_predictions.head()
```

Out[43]:

	Dates	LR	SVR	Holts Linear Model Prediction
0	2020-04-25	1560529	3322586	2855246

project - Jupyter Notebook

localhost:8888/notebooks/190050079/project.ipynb

jupyter project Last Checkpoint: 10/22/2021 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

In [41]:
```
model_train=datewise.iloc[:int(datewise.shape[0]*0.85)]
valid=datewise.iloc[int(datewise.shape[0]*0.85):]
```

In [42]:
```
holt=Holt(np.asarray(model_train["Confirmed"])).fit(smoothing_level=1.4,smoothing_slope=0.2)
y_pred = valid.copy()
y_pred["Holt"]=holt.forecast(len(valid))
```

In [43]:
```
holt_new_date=[]
holt_new_prediction=[]
for i in range(1,18):
   holt_new_date.append(datewise.index[-1]+timedelta(days=i))
   holt_new_prediction.append(holt.forecast((len(valid)+i))[-1])

model_predictions["Holts Linear Model Prediction"]=holt_new_prediction
model_predictions.head()
```

Out[43]:

	Dates	LR	SVR	Holts Linear Model Prediction
0	2020-04-25	1560529	3322586	2855246
1	2020-04-26	1582219	3500761	2933902
2	2020-04-27	1603909	3686599	3012558
3	2020-04-28	1625599	3880344	3091214
4	2020-04-29	1647289	4082245	3169870

In [ ]:

Conclusion & Future Scope:

In machine learning there are two types one is supervised learning and unsupervised learning. Supervised learning builds the model which makes the predictions based on the input and the output. Unsupervised learning develops the model from the input data alone.

Outline: Here we are going to use the SVM model and linear regression method to predict the outbreak of coronavirus for the upcoming 10 days across different regions by using charts and graphs.

Coronavirus doesn't have the ability to mobilize themselves from one host to another host. But it can able to multiply themselves once it gets into a host. So by considering the above scenario, we can come to a conclusion that it spreads via the physical network.

A machine learning model has been developed to predict the estimation of the spread of the COVID-19 infection in many countries and the expected period after which the virus can be stopped. Globally, our results forecasted that the COVID-19 infections will greatly decline during the first week of September 2021 when it will be going to an end shortly afterward. Moreover, we can apply our proposed model to other countries that are affected by the COVID-19. Additionally, our model could also evaluate the effect of the public health guidelines, infection control, and lock-down decisions that were taken to stop the COVID-19 pandemic. Future work could focus on applying a deep learning model by using big data as training data. Moreover, our proposed model can apply to specific countries.

References:

1.Wang P.W., Horby F.G., Hayden G.F. GaoA novel coronavirus outbreak of global health concern. *Lancet.* 2020;395(10223):470–473. [PMC free article] [PubMed] [Google Scholar]

2. S. Tuli, S. Tuli, G. Wander, P. Wander, S.S. Gill, S. Dustdar, R. Sakellariou, O. Rana, Next generation technologies for smart healthcare: challenges, vision, model, trends and future directions, Internet Technol. Lett.e145.

3. B.V.V.S. Narayana K.S. Ravi N.V.K. Ramesh, (2018),'A review on advanced crop field monitoring system in agriculture field through top notch sensors', J. Adv. Res. Dyn. Contr. Syst.,10(6 Special Issue),PP. 1572- 1578

4. S. Sridevi P.M. Bindu P.V.S.J. Krishna Teja , (2018),'User behavior analysis on agriculture mining system', Int. J. Eng. Technol. (UAE),7(2),PP. 37- 40

5. Y. Bai, L. Yao, T. Wei, F. Tian, D.-Y. Jin, L. Chen, M. WangPresumed asymptomatic carrier transmission of COVID-19 JAMA (2020) [PMC free article] [PubMed]

6. D. Radhika K.D. Aruna The smart triad: Big data analytics, cloud computing and internet of things to shape the smart home, smart city, smart business & smart country International Journal of Recent Technology and Engineering, 8(2 Special Issue 11) 2019 PP.3594-3600

7. Kalavala S.S., Sakhamuri S., Prasad B.B.V.S.V. (2019), 'An efficient classification model for plant disease detection', International Journal of Innovative Technology and Exploring Engineering, 8(7), PP.126-129.

8. Likitha K., Aruna Sri P.S.G., Ratna Phanitha K. (2017),'A secure data sharing and revocation in cloud using IDE',International Journal of Applied Engineering Research,12(Special Issue 1),PP.228-235.

9. S. Sakhamuri A. Virupakshi V. Pushpalatha D. Nagamani 'Elimination of redundant data in cloud with secured access control', International Journal of Innovative Technology and Exploring Engineering 8 6 2019 PP.1344-1348

10. Y. Zhang, B. Jiang, J. Yuan, Y. TaoThe impact of social distancing and epicenter lockdown on the COVID-19 epidemic in mainland China: a data-driven SEIQR model study medRxiv (2020)